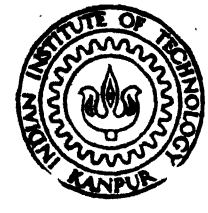
CONNECTIONIST ARCHITECTURE AS A MODEL OF LANGUAGE ACQUISITION

by

SUDHIR KUMAR



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

AUGUST, 1988

CONNECTIONIST ARCHITECTURE AS A MODEL OF LANGUAGE ACQUISITION

A Thesis Submitted

In Partial Fulfilment of the Requirements

for the Degree of

MASTER OF TECHNOLOGY

by

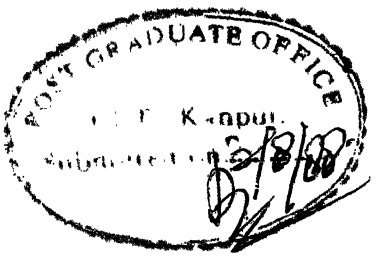
SUDHIR KUMAR

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

AUGUST, 1988



C E R T I F I C A T E

This is to certify that the thesis entitled "CONNECTIONIST ARCHITECTURE AS A MODEL OF LANGUAGE ACQUISITION" is a report of the work carried under my supervision by SUDHIR KUMAR, and that it has not been submitted elsewhere for degree.

S. Biswas.

(S. Biswas)

Dept of Computer Science

& Engineering

I.I.T. Kanpur

ABSTRACT

We usually have implicit knowledge of grammatical rules. Traditional view about how exactly the rules are stored in our brain is that the rules are stored in explicit form as propositions. These propositions cannot be described verbally because they are sequestered in such a way that only the language processing unit can access.

An alternative to this view, based on **Parallel Distributed Processing** model has been proposed by Rumelhart and McClelland. We have explored this model in the context of formation of past tense forms from present tense ones. The model has also been used for capturing the notion of sandhi.

ACKNOWLEDGEMENT

I would like to express my deep sense of gratitude to Dr. Biswas and Dr. Karnick for the constant help, guidance and encouragement they have given to me. I also tender my sincerest apologies for all my carelessness which Dr. Biswas very patiently condoned.

I thank S. Khadilkar for providing me moral support and offering me all possible help. I thank Ajay Pandit to have let me use his PC and printer inspite of his own inconviences.

August, 1988

Sudhir Kumar

CONTENTS

C H A P T E R 1

INTRODUCTION

Though we all make mistakes when we speak, we have a pretty good ear for what is wrong and what is right -- and our judgments of correctness or grammaticality can be characterized by rules. Therefore, in some sense, we know the rules of our language. However, our knowledge of rules is implicit as we need not necessarily be able to state the rules explicitly.

There can be two views about the characterization of this implicit knowledge. One view, commonly known as **explicit inaccessible rule view**, holds that the rules of the language are stored in explicit form as propositions, and are used by language comprehension, production and judgment mechanisms. These propositions cannot be described verbally only because they are sequestered in a specialized subsystem which is used in the language processing, or because they are written in a special code that only the language processing unit can understand.

The other view, based on **Parallel Distributed Processing** (PDP) model holds that the mechanisms, which process language and make judgments of grammaticality are constructed in such a way that their performance is characterized by rules, but the rules themselves are not written in explicit form anywhere in the mechanism. D.E.Rumelhart and J.L.McClelland [1986] tried to simulate a simple but realistic phenomenon viz. morphisms in the English verbs from the present tense form to the past tense form

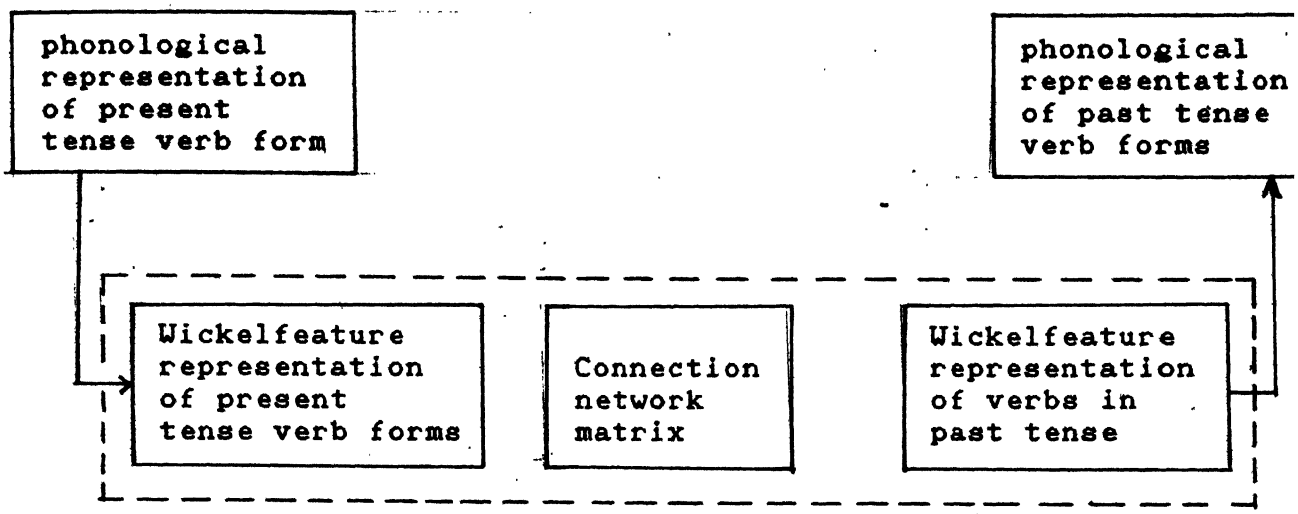
by the PDP model. They reported very favourable results in support of the PDP model (PDP model is also called network model or connectionist model as it works through a connection network between input units and output units).

Subsequently, Pinker and Prince [1988] published a paper criticizing the model and reported failures of the model on many counts. They also pointed that the failures of the model can be attributed to its connectionist architecture thus suggesting that the explicit inaccessible rule view offers a better model for learning of language.

Inspite of objections raised by Pinker & Prince (some of which seem to be serious), the idea of connectionist model looks intuitively appealing. This led us to repeat the RM experiment to obtain an insight into the model.

First of all we went through the entire simulation which Rumelhart & McClelland had performed, viz. behaviour of the model for simulating changes in the verbs from the present tense form to the past tense form. We studied the behaviour of the model very carefully for different sets of inputs and tried to check whether the behaviour of the model matches with the real life experience.

For this simulation, the following scheme was used:



(Details of the simulation are given in the following chapters)

We tried to evaluate the model in the light of objections raised by Prince & Pinker [1988]. In particular, we endeavoured to explore whether the failures of the model are the result of the connectionist architecture or whether they can be ascribed to the coding scheme used to represent words.

Results we obtained from the simulation suggest that the failures of the model can be ascribed to the coding scheme instead of the connectionist architecture.

As the changes in words to form new words (e.g. words in different parts of speech from the same word; combining two words to form a new word etc.) is a phenomenon common to all the languages, the model can be tested to capture morphisms in different contexts.

We also experimented simulating 'sandhi rules' through the PDP model. The model's response was very accurate.

Rest of the thesis is organized as follows:

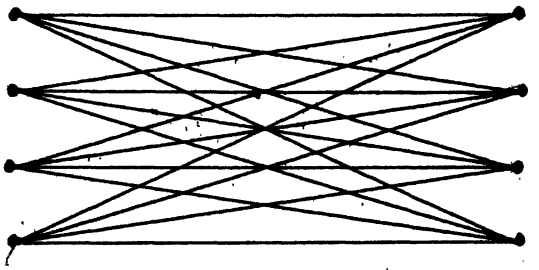
In chapter 2, we discuss the PDP model; in chapter 3, details of simulations we performed for the learning of the past tense; in chapter 4 results of the simulation; in chapter 5, simulation of sandhi and in chapter 6, criticism (for and against) of the model.

C H A P T E R 2

DESCRIPTION OF THE PDP MODEL

2.1 PATTERN ASSOCIATOR NETWORK:

Consider a pattern associator network as described below:



pattern associator network

FIG 2.1

Dots on the left side represent input units and dots on the right side represent output units. There are weighted connections between the input units and the output units.

For any given set of inputs, the model computes, for each output unit, the net input to it from all the weighted connections from the input units. Algebraically, the net input to output unit j is

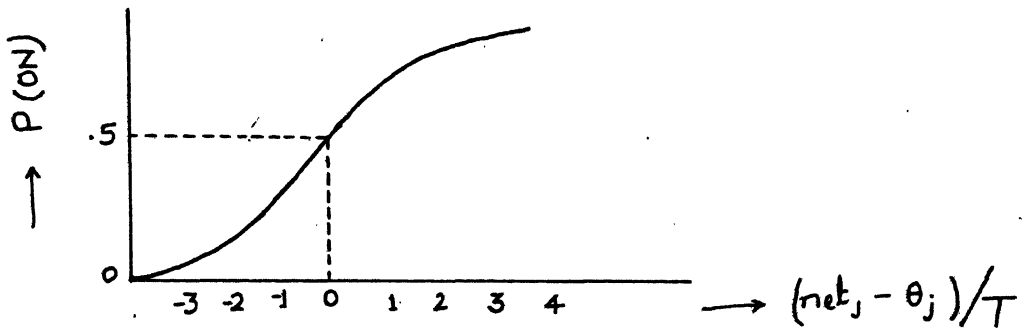
$$net_j = \sum a_i w_{ij}$$

where a_i represents the activation of input unit i and w_{ij} represents the weight from unit i to unit j. Each output unit has a threshold θ, which is adjusted by a learning procedure that we

will describe shortly. The probability that the unit is turned on depends on the amount the net input exceeds the threshold. Logistic probability function to determine whether the unit is turned on is given by :

$$p(a_j = 1) = \frac{1}{1 + \exp(-(net_j - \theta_j)/T)}$$

where T represents the temperature of the system, a parameter to be chosen suitably. At very high temperatures, the response of the units is highly variable; with lower values of T, the units behave more like linear threshold units. The logistic function is shown in the figure below :



logistic probability function

FIG 2.2

We focus our attention on the application of such a pattern association mechanism for representing rules for mapping one set of patterns into another.

2.2 LEARNING:

On a learning trial the, model is presented with both the input pattern and the target output pattern. The pattern

associator network computes the output it would generate from the input. Then for each output unit, model compares its answer with the target. Then it adjusts the connection using the perceptron convergence procedure. The exact procedure is as follows:

When the computed output matches the target output, the model is doing the right thing and so none of the weights on the line coming into the unit are adjusted. When the computed output is 0 and the target says it should be 1, we want to increase the probability that the unit will be active the next time the same input pattern is presented. To do this, we increase the weights from all the input units that are active by a small amount η. At the same time, the threshold is also reduced by η. When the computed output is 1 and the target says it should be 0, we want to decrease the probability that the unit will be active the next time the same input pattern is presented. To do this, the weights from all the input units that are active are reduced by η and the threshold is increased by η.

It is expected that after sufficient number of learning trials, the network receives connection strengths appropriate for mapping a number of different input patterns to a number of different output patterns. The perceptron convergence procedure can accommodate a number of arbitrary associations between input patterns and output patterns as long as input patterns form a linearly independent set RM[1986].

The restriction of networks such as this to linearly independent sets of patterns is a severe one since there are only

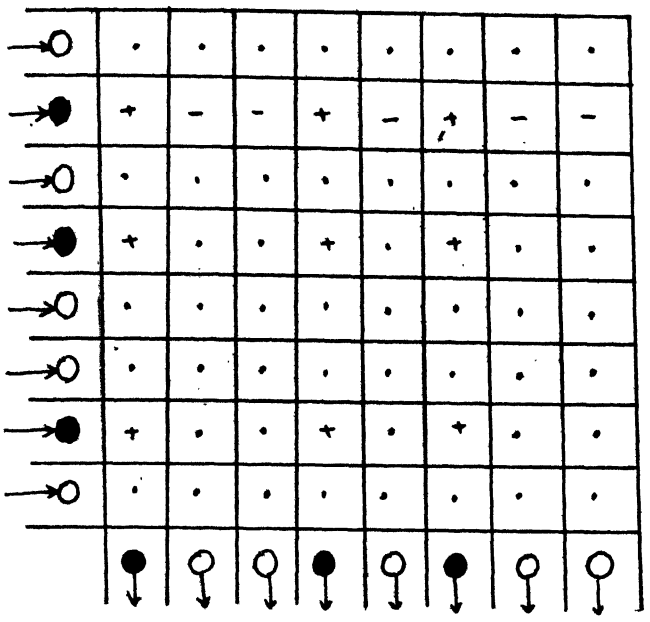
N linearly independent patterns of length N. That means we could store at most N unrelated associations in the network of size N and maintain accuráte performance. However, if the patterns conform to certain general rules, the capacity of the network can be greatly enhanced.

For example, we will see that in the demonstration model of size 8 (to be described shortly), the set of connections obtained through the perceptron convergence procedure is good enough to give correct mapping for all the 18 different input patterns.

2.3 ILLUSTRATION:

To illustrate the model, we use a simple network of eight input and eight output units and a set of connections from each input unit to each output unit. The rules dictate the transformation of the triplets of inputs unit to the triplets of output units. The network is illustrated in the figure 2.3.

Next to the network is the matrix of connections abstracted from the actual network itself. Thus entry in the i^{th} row of the j^{th} column indicates the connection w_{ij} from the i^{th} input unit to the j^{th} output unit. Using this diagram, it is easy to compute the net inputs that will arise on the output units when an input pattern is presented. For each output unit, one simply scans across its column and adds all the weights found in the rows associated with active input units. For the weights given in the figure, it can easily be verified that when the input pattern



0	0	0	0	0	0	0	0
15	-18	-16	15	-19	17	-16	-18
0	0	0	0	0	0	0	0
15	-18	-16	15	-19	17	-16	-18
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
15	-18	-16	15	-19	17	-16	-18
0	0	0	0	0	0	0	0

Simple n/w used in illustrating basic properties of pattern associator. The darkened units are the active units.

Fig 2.3

illustrated on the left hand panel is presented, each output pattern that should be on, receives a high net weight and consequently high probability of turning on.

On page 11, Table 2.1, we present the matrix of connection strengths for various mappings of input patterns to output patterns.

In table 2.1D, we present the matrix of connection strengths acquired for the rule of 78, an example originally used for illustration by Rumelhart and Mc-Clelland [1986].

Rule of 78

---Input is triplets of one active unit from each of the following set

(1,2,3) (4,5,6) (7,8)

--- Output pattern paired with a given input pattern consists of

the same unit from (1,2,3)
the same unit from (4,5,6)
the other unit form (7,8)

examples:

(2,4,7) --> (2,4,8); (1,6,7) --> (1,6,8)

(3,5,8) --> (3,5,7); (3,6,7) --> (2,6,8) etc.

exception:

(1,4,7) --> (1,4,7)

We see that the matrix of size 8 gives accurate performance for 18 different patterns.

In the following paragraphs, we quote some observations of the way children learn morphisms of the present tense verb forms to the past tense verb forms. Then we will show how the pattern associator network model is able to demonstrate similar behaviour, hence suggesting that it can be reasonable alternative to explicit inaccessible rule model.

2.4 THE PHENOMENON:

Brown[1973], Erwin[1964] and Kuczaj[1977] report a sequence of 3 stages in the acquisition of the use of past tense by children learning English as their native tongue.

TABLE 2.1

Weights in 8 unit network after various learning experiences

A.Weights acquired in learning (3,4,6)--> (3,6,7)

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
-17	-18	17	-16	-19	17	15	-18
-17	-18	17	-16	-19	17	15	-18
0	0	0	0	0	0	0	0
-17	-18	17	-16	-19	17	15	-18
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Threshold:

17 18 -17 16 19 -17 -15 18

B. Weights acquired in learning (2,4,7)-->(1,4,6)

0	0	0	0	0	0	0	0
15	-18	-16	15	-19	17	-16	-18
0	0	0	0	0	0	0	0
15	-18	-16	15	-19	17	-16	-18
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
15	-18	-16	15	-19	17	-16	-18
0	0	0	0	0	0	0	0

Threshold:

-15 18 16 -15 19 -17 16 18

C.Weights acquired in earning A and B together

0	0	0	0	0	0	0	0
23	-12	-21	19	-12	12	-20	-13
-22	-11	22	-19	-12	9	20	-11
1	-23	1	0	-24	21	0	-24
0	0	0	0	0	0	0	0
-22	-11	22	-19	-12	9	20	-11
23	-12	-21	19	-12	12	-20	-13
0	0	0	0	0	0	0	0

Threshold:

-1 23 -1 0 24 -21 0 24

D.Weights acquired in learning rule of 78

55	-30	-28	-3	-6	0	12	-16
-31	50	-28	-3	-2	-5	-2	3
-32	-27	47	-2	-1	-5	-7	6
-2	-5	-7	51	-31	-32	14	-16
-4	-1	-1	-29	53	-30	-5	2
-2	-1	-1	-30	-31	52	-6	7
-4	-4	-3	-3	-6	-3	-38	36
-4	-3	-6	-5	-3	-7	41	-43

Threshold:

8 7 9 8 9 10 -3 7

---

a In stage 1, children use only a small number of verbs in the past tense. These are very high frequency words and majority of these are irregular. At this stage, children tend to get the past tenses of these words correct if they use the past tense at all. For example, a child's vocabulary of past tense verbs might consist of -- came, got, gave, looked, needed, took and went. Of these seven verbs, only two are regular and other five are irregular. At this stage, there is no evidence of the use of rule -- it appears that children simply know a small number of separate items.

b In stage 2, children use a much larger number of verbs in the past tense. These verbs include a few more irregular verbs, but it turns out that the majority of the words at this stage are regular. Also, evidence of implicit knowledge of linguistic rule emerges -- as there are two crucial facts in support of this --

.the child can now generate a past tense for an invented word.

.children now incorrectly supply regular past tense endings for words which they used correctly in stage 1. These errors may involve either adding ed to the roots as in comed /k^md/ or adding ed to the irregular past tense form as in camed /kAmd/

Such findings have been taken as fairly strong support for the assertion that the child at this stage has acquired the past tense rule.

c In stage 3, the regular and irregular forms coexist. That is, children have regained the use of correct irregular forms of the past tense, while they continue to apply the regular form to new words they learn.

Let us now consider the case for the pattern associator like the situation a young child faces in learning the pas tenses of English verbs.

The illustrative 8 unit model was first presented with tw pattern pairs. One of these was a regular example of 78-rule viz. [(2,5,8) --> (2,5,7)], the other was an exception to th rule, [(1,4,7) --> (1,4,7)]. The exception is analogous to th

irregular verbs. The simulation saw both the pairs for 25 times and the connection strength was adjusted after each presentation.

The resulting set of connections is shown in table 2.2A. These many presentations are sufficient for giving almost perfect performance for both the patterns. That is it has learnt a set of connections that can accommodate these two patterns, but it cannot generalize to new instances of the rule. This situation corresponds to stage 1 of the learning by a child.

But as the child learns more and more verbs, the proportion of regular verbs increases. This changes the situation for the learning model. Now the model is faced with a number of examples most of which follow the rule. This new situation changes the experience of the network and thus the pattern of interconnection it contains. Because of the predominance of the regular form in the input, the network learns the regular pattern, temporarily over regularizing exceptions that it may have previously learnt.

In the illustration, for the second stage of learning, we present the model with the entire set of eighteen input patterns. At the end of 10 exposures to the full set of 18 patterns, the model has learnt a set of connection strengths that predominantly captures the regular pattern. At this point, its response to the exceptional pattern is worse than it was before the beginning of phase 2. Rather than getting the right output for Units 7 and 8 the system is now regularizing it.

The reason for this behaviour is very simple. All that is

TABLE 2.2

Weights in 8 unit network after various learning experiences

A. weights acquired in learning
(2,5,8) --> (2,5,7)
(1,4,7) --> (1,4,7)

16	-18	-11	16	-17	-12	12	-13
-16	12	-9	-13	13	-10	9	-12
0	0	0	0	0	0	0	0
16	-18	-11	16	-17	-12	12	-13
-16	12	-9	-13	13	-10	9	-12
0	0	0	0	0	0	0	0
16	-18	-11	16	-17	-12	12	-13
-16	12	-9	-13	13	-10	9	-12

Threshold:

0	6	20	-3	4	22	-21	25

B. weights acquired after
10 more exposures to
all the 18 associations

42	-30	-18	5	-13	-7	12	-13
-28	36	-20	-8	7	-7	0	2
-16	-16	31	-2	-3	4	-7	6
10	-11	-6	39	-31	-21	9	-12
-11	2	-1	-23	40	-18	0	1
-1	-1	0	-21	-18	29	-4	6
6	-11	-4	5	-12	-5	-22	22
-8	1	-3	-10	3	-5	27	-27

Threshold:

2	10	7	5	9	10	-5	5

C. weights acquired after 30 more
exposures to all the 18
associations

63	-35	-27	0	-7	-2	17	-18
-38	59	-31	-5	1	-3	-5	4
-34	-32	49	-4	-3	-2	-7	9
2	-6	-7	59	-40	-36	22	-17
-11	-2	-1	-34	61	-30	-6	3
0	0	-1	-34	-30	59	-11	9
-4	-5	-5	-4	-6	-2	-43	42
-5	-3	-4	-5	-3	-5	48	-47

Threshold:

9	8	9	9	9	7	-5	5

D. weights acquired after
100 more exposures to
all the 18 associations

88	-48	-44	-3	-3	-1	38	-36
-52	84	-46	-3	-2	-2	-14	12
-45	-48	79	-5	-7	-7	-16	16
1	-5	-6	82	-54	-47	36	-37
-5	-5	-2	-48	82	-44	-13	14
-5	-2	-3	-45	-40	81	-15	15
1	-4	-8	-4	-7	-6	-63	60
-10	-8	-3	-7	-5	-4	71	-68

Threshold:

9	12	11	11	12	10	-8	8

happening is that the model is continually being bombarded with the learning experiences directing it to learn the rule of 78. On only one learning trial out of 18, is it exposed to on exception to this rule.

At the end of 10 cycles, we can see that the model is building up extra excitatory connection from input unit 1 and 4 to output unit 7 and extra inhibitory strength from unit 1 & 4 to unit 8, but these are not strong enough to make the model get the right answer for output units 7 & 8 when (1,4,7) input pattern is shown.

The situation is analogous to stage 2.

It is only after the model has reached the stage where it is making very few mistakes on the 17 regular patterns that it begins to accommodate the exception. This amounts to making the connection from units 1 & 4 to output unit 7 strongly excitatory and making the connections from these units to output unit 8 strongly inhibitory. Finally in table 2.2D, after a large numbe of cycles through the entire set of 18 patterns, the weights are sufficient to get the right answers nearly all the time. This situation can be thought as analogous to stage 3.

Of course, the example we have considered in this section is highly simplified. However, it illustrates several basic facts about the pattern associators :

> They tend to exploit regularity that exists in the mapping from one set of patterns to another. Indeed, this is one of the main advantages of the use of distributed representations.

They allow exceptions and regular patterns to coexist in the same network.

If there is predominant regularity in a set of patterns this can swamp exceptional patterns until the set of connections has been acquired that captures the predominant regularity. Then further gradual tuning can occur that adjusts these connections to accommodate both the regular pattern and the exceptions.

These basic properties of the pattern associator model lie at the heart of the 3-stage learning process and, account for the gradualness of the transition from stage 2 to stage 3.

CHAPTER 3

DETAILS OF SIMULATION FOR LEARNING THE PAST TENSE

3.1 INTRODUCTION :

The preceding chapter describes basic aspects of the behaviour of the pattern associator model and explains what happens when a pattern associator is applied to the processing of English verbs following a training schedule similar to the one we have considered for the acquisition of rule of 78.

For actual simulation of the pattern associator for processing of English verbs the base form of the verbs and the correct past tenses of these verbs must be represented such that the features provide a convenient basis for capturing the regularities embodied in the past tense forms of English verbs. Basically, there are two considerations:

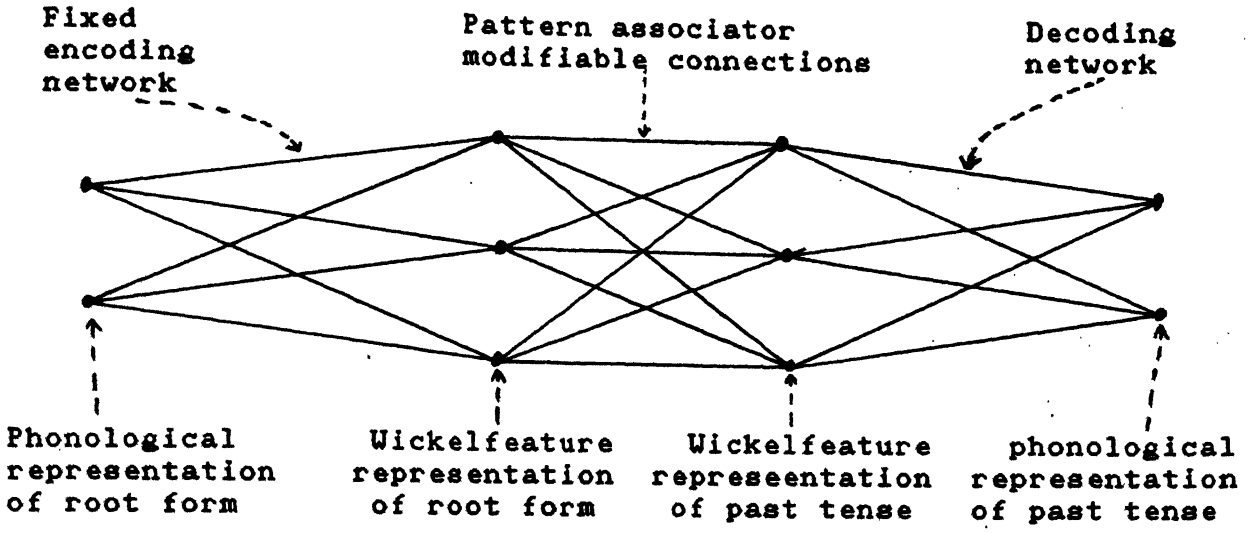
.A representation is needed that would permit differentiation of all of the root forms of English and their past tenses.

.A representation is needed that would provide a natural basis for generalizations to emerge about what aspects of a present tense correspond to what aspects of the past tense.

It is difficult to find a scheme which meets the second criterion. RM used the Wickelfeature representation of the verbs and their past tenses. This representation more or less meets the

first criterion.

The basic structure of the model is illustrated below:



Basic structure of the model

FIGURE 4

3.2 WICKELFEATURE REPRESENTATION :

The basis for Wickelfeature representation is a scheme proposed by Wickelgren [1969]. He suggested that words should be represented by context sensitive phoneme units which represent each phone in a word as a triple, consisting of the phone itself, its predecessor and its successor. We call these triples **Wickelphones**. A phoneme occurring at the beginning of a word is preceded by a special symbol (#) standing for the word boundary; likewise a phoneme occurring at the end of a word is followed by #. The word cat (/kat/) for example would be represented as #ka, kat, at# and the word subscribe(/s^bskrIb/) would be represented

by #s^, s^b, ^bs, bsk, skr, krI, rIb, Ib#.

Though the Wickelphones in a word are not exactly position specific, it turns out that

a. few words contain more than one occurrence of any given Wickelphone

b. there are no two words we know of that consist of the same sequence of Wickelphones. For example /slit/ and /silt/ contain no Wickelphone in common.

One nice property of Wickelphones is that they capture enough of the context in which a phoneme occurs to provide a sufficient basis for differentiating between the different cases of the past tense rule and for characterizing the contextual variables that determine the subregularities among the irregular past tense verbs. For example, the word final phoneme that determine whether we should add /d/, /t/ or /^d/ in forming the regular past. And it is the sequence iN# which is transformed to aN# in the ing --> ang pattern found in the words like sing --> sang, cling --> clang etc.

The trouble with Wickelphone solution is that there are too many of them. Assuming that 35 different phonemes are distinguished the number of Wickelphones would be 35*35*35 even without counting Wickelphones containing word boundaries.

Obviously a more compact representation is required. This can be obtained by representing each Wickelphone as a distributed pattern of activation over a set of feature detectors. The basic

lea is that each phoneme is represented not by a single ickelphone but by a pattern of what is called **Wickelfeatures.** ach Wickelfeature is a conjunctive, or a context sensitive eature capturing a feature of the central phoneme, a feature of he predecessor and a feature of the successor.

.3 DETAILS OF WICKELFEATURE REPRESENTATION :

First we describe the simple feature representation scheme ised for coding a single phoneme as pattern of features without egard to its predecessor and successor. Then we describe how this scheme is extended to code whole Wickelphones.

To characterize each phoneme, a highly simplified feature set illustrated in TABLE 3.1 is used. The purpose of the scheme is:

a. to give as many of the phonemes as possible, a distinctive code

b. to allow code similarity to reflect the similarity structure of the phonemes in a way that seemed sufficient for present purpose.

c. to keep the number of different features as small as possible.

The coding scheme can be thought of as categorizing each phoneme on each of four dimensions. The first dimension divided the phonemes into 3 major types: interrupted, continuous consonants (fricatives, liquids and semi vowels), and Vowels(high

and low). The second dimensions further subdivides these major classes, e.g. vowels into high and low, continuous consonants into fricatives and sonorants (liquids and semi vowels lumped together). The third dimension classifies the phonemes into three rough places of articulation -- front, middle and back. The fourth dimension subcategorizes the consonants into voiced vs. unvoiced and vowels into long and short.

TABLE 3.1

Categorization of phonemes on 4 simple dimensions

	Place					
	front		middle		Back	
	V/L	U/S	V/L	U/S	V/L	U/S
Interrupted						
stop	b	p	d	t	g	k
nasal	m	--	n	--	N	--
Cont. Consonant						
fricatives	v/D	f/T	z	s	Z/j	S/C
liquids/semi vowels	w/l	--	r	--	y	h
Vowel						
high	E	i	O	x	U	u
low	A	e	I	a/X	W	q/o

Key N = ng in sĭng; D = th in the; T = th in with; S = sh in ship Z = z in azure; C = ch in chip; E = ee in beet; i = i in bit; O = oa in boat; x = u in but; X = a in father; U = oo in boot; u = oo in book; A = ai in bait; e = e in bet; I = i_e in bite; a = a in bat; W = ow in cow; q = aw in saw; o = o in hot.

Using the above code each phoneme can be characterized by one value on each dimension. If a unit is assigned for each value on each dimension, 10 units would be needed to represent the features of a single phoneme since two dimensions have 3 values and two have 2 values.

Wickelphones are represented by sets of triplets of features called Wickelfeatures. Each triplet contains one feature from central phoneme, one from predecessor phoneme and one from the successor phoneme. In each Wickelfeature, values of predecessor and successor phoneme features are always taken on the same dimension. Each Wickelphone will turn on 16 Wickelfeature detectors. In table 3.2 we present the list of 16 Wickelfeatures for the Wickelphone **/kAm/**.

The first Wickelfeature is turned on whenever there is a Wickelphone in which the preceding contextual phoneme is an interrupted consonant, central phoneme a vowel and the following phoneme is an interrupted consonant. The same Wickelfeature would be turned on for **bid**, **p^l**, **map** and many other Wickelphones.

Now words are simply lists of Wickelphones. Thus they can be represented by simply turning on all of the Wickelfeatures in any Wickelphone of a word. In all, there are 460 Wickelfeatures. Hence all words, no matter how many phonemes are there in the word, will be represented by a subset of these 460 Wickelfeatures.

TABLE 3.2

SIXTEEN WICKELFEATURES FOR THE WICKELPHONE kAm

Feature	Preceding context	Central phoneme	Following context
1.	Interrupted	Vowel	Interrupted
2.	Back	Vowel	Front
3.	Stop	Vowel	Nasal
4.	Unvoiced	Vowel	Voiced
5.	Interrupted	Front	Vowel
6.	Back	Front	Front
7.	Stop	Front	Nasal
8.	Unvoiced	Front	Voiced
9.	Interrupted	Low	Interrupted
10.	Back	Low	Front
11.	Stop	Low	Nasal
12.	Unvoiced	Low	Voiced
13.	Interrupted	Long	Vowel
14.	Back	Long	Front
15	Stop	Long	Nasal
16.	Unvoiced	Long	Voiced

Although the model is not completely immune to th possibility that two different words will be represented by th same pattern, we encountered little difficulty in decoding any o

the verbs we studied.

3.4 SUMMARY OF THE STRUCTURE OF THE MODEL:

In summary, the model contains two sets of 460 Wickelfeature units, one set (input units) to represent the base form of each verb and one set (output units) to represent the past tense form of each verb.

The model is tested by typing in an input phoneme string which is translated by the fixed encoding network into a pattern of activation over the set of input units (Wickelfeatures). Each active input unit contributes to the net input of each output by an amount and direction (+ve or -ve) determined by the weight on the connection between the input unit and the output unit. The output units are then turned on or off probabilistically, according to the logistic activation function mentioned in chapter-2. The output pattern generated this way is decoded to get the phonological representation of the past tense form.

The model is trained by providing it with pairs of patterns consisting of base pattern and target, or correct, output. It compares what it generates internally to the target output, and when it gets the wrong answer for a particular output unit, it adjusts the strength of connection between input and output unit so as to reduce the probability that it will make the same mistake the next time the same input pattern is presented.

In the logistic probability function

$$p(a_j = 1) = \frac{1}{\exp(-(net_j - \theta_j)/T)}$$

when a low value of T is used, the system is linear, whereas when a high value of T is used system's response is highly variable. It turns out that at higher values of T, learning is relatively fast. We used the value T=200 for learning. For adjusting weights of connections and threshold , we used =1.

3.5 THE SIMULATION:

In accordance with a child's learning experience, the model was trained for 10 highest frequency verbs for 25 cycles. The 10 verbs were -- come, get, give, look, take, go, have, live feel and make. It learnt all the above mentioned verbs perfectly. Note that 8 out of 10 verbs are irregular.

We take the performance of the model at this point correspond to the performance of a child in phase 1 of acquisition.

To simulate later phases of learning the system was given around 200 learning trials on 200 verbs (190 new verbs added to the earlier 10 verbs). Each trial consisted of one presentation of each of 200 verbs. The responses of the model early on in this phase of training correspond to phase 2 of the acquisition process; its ultimate performance at the end of 200 exposures to each of 200 verbs correspond to phase 3. At this point, model exhibits almost errorless performance on the basic 200 verbs, Finally new verbs were presented to the system and the transfer responses to these were recorded. During this stage, connection strengths were not adjusted. Performance of the model in various phases is discussed in the next chapter.

3.6 DECODING NETWORK :

(From Wickelfeatures to phonological representation)

We assign weights to Wickelphones as follows:

Suppose a Wickelfeature, F_i, can be activated by Wickelphones P_{ji}. Then to the weight of each Wickelphone P_{ji}, we a weight $1/(P_{ji})$.

i.e. total weight received by any Wickelphone P_m

$$\text{where } P_{ji} = \begin{cases} 1 \text{ if Wickelphone } P_m \text{ activates Wickelfeature } F_i \\ 0 \text{ otherwise} \end{cases}$$

EXAMPLE

To form past tense form of the verb give /giv/, the model computes that the following Wickelfeatures should be active in the past tense form:

2 3 6 9 64 68 80 89 104 123 127 133 150 156 160 169 172 173 179
181 232 233 236 239 278 279 282 285 288 294 298 307 310 311 317
319 370 371 374 377 386 390 402 411 426 445 449 455

Now the Wickelfeature number 2 can be activated by the following Wickelphones:

#bE #bA #bi #gA #gu #gW #pa #pX .....

Total number of all such Wickelphones is 46. Hence all those Wickelphones receive a weight of 1/46.

Similarly, Wickelphones are assigned weights for other Wickelfeatures also.

After assigning weights for all the Wickelfeatures in th

above list, we have the following situtation:

Wickelfeatures	total weight
Av#	.472
AD#	.472
Aw#	.472
Al#	.472
#gA	.358
gAv	.247
gAw	.247
gAD	.247
gAl	.247
#gE	.176

Now the Wickelphone receiving larger weight is likely to be present in the phonological representation of the verb,as that activates large no. of Wickelfeatures present in Wickelfeature representation of the verb. Hence choose that Wickelfeature for the phonological representation.

In case of more than one wickelphone having comparable weights, choose the one whose central phoneme is consonant appearing in the present tense form of the verb. For example, while decoding the set of Wickelfeature computed by the network for the past tense of drive /drIv/, suppose Wickelphones Ov# and OD# receive comparable weights. Then Ov# will be chosen in preference to OD#, as the central phoneme v in Ov# is a consonant appearing in the present form of drive.

After choosing a Wickelphone, update the set of active Wickelfeatures as follows:

{set of updated active Wickelfeatures} =
{active Wickelfeatures} - {Wickelfeatures which ca
be activated by the chosen Wickelphones }

Example contd.

We find that 4 Wickelphones receive equal weights. From these, choose the one whose central phoneme is a phoneme occurring in the present tense form. Hence the Wickelphone to be chosen will be **Av#**.

Wickelphone Av# activates the Wickelfeatures

64 68 80 89 156 160 172 181 294 298 310 319 386 390 402 411

Hence after choosing the Wickelphone Av#, update the set of active Wickelfeatures by removing all the Wickelfeatures activated by Av#.

Therefore, the set of activated Wickelfeatures =

{ 2 3 6 9 104 123 127 133 150 169 173 179 232 233 236 239 278 279
282 285 288 307 311 317 370 371 374 377 426 445 449 455 }

Repeat the process of choosing a Wickelphone with the new set of active Wickelfeatures. At the end we find that two other Wickelphones which are chosen are #gA and gAv.

The three Wickelphones can be combined to form the past tense form, in this case /gAv/.

The process of choosing Wickelphones is repeated till they can be arranged to form a word close to the past tense form of the verb.

The decoding process was carried out interactively to guide the system fill out the Wickelphones it missed occasionally, to form the past tense form.

To evaluate the system response, the set of Wickelfeatures activated by decoded word is compared to the set of Wickelfeatures computed by the system from present tense form. It was found that most often the tally was closest when the decoded word was the past tense.

Results of the simulation is discussed in the next chapter.

CHAPTER 4

RESULTS OF SIMULATION

To evaluate the system's performance, we need to feed it verbs in their present tense form. The model then, should find out the Wickelfeatures activated by the verb, from them compute the Wickelfeatures for the past tense form and decode the Wickelfeatures to find out the phonetic representation of the past tense form. Then we should try to compare the model's response for the past tense form to the actual past tense form.

But the process of decoding Wickelfeatures to find out phonetic representation is extremely slow. Hence a different strategy, which is described below, is used to evaluate the system's performance.

For every verb, we give the model many alternatives for the past tense form. For example, for the verb **come /k^m/**, the model is given the alternatives **came /kAm/**, **camed /kAmd/** and **comed /k^md/**.

Given a verb, model computes the Wickelfeatures which should be active in its past tense form. Then for each alternative for past tense of that verb, it computes the Wickelfeatures the alternative will turn on and finds the percentage match in the Wickelfeatures.

$$\text{percentage match} = \frac{\text{matches}}{\text{matches} + \text{misses} + \text{false alarms}} * 100$$

where

matches = number of matching Wickelfeatures

miss = Wickelfeatures which are computed to be in the past tense form but are not turned on by the alternative.

false alarm = Wickelfeatures which are turned on by the alternative but are not active in the computed Wickelfeatures.

For example, to compare the alternatives came, camed, and comed, as past tense for come:

from come, the model, using the connection matrix, computes that the following Wickelfeatures should be active,

2 3 4 6 9 10 22 34 43 103 123 130 134 149 156 160 169 172 180 181
194 195 222 226 227 232 233 234 236 239 287 294 298 304 310 314
318 319 324 326 329 331 333 370 371 372 374 375 377 378 379 432
436 445 448 452 456 457

Wickelfeatures which are turned on by came /kAm/ are:

2 3 6 18 22 34 43 103 123 130 134 149 156 160 169 172 176 180 181
232 233 236 239 287 294 298 307 310 314 318 319 324 325 328 331
370 371 374 377 425 432 436 445 448 452 456 457

Hence, for came /kAm/

missing Wickelfeatures are

4 9 10 194 195 222 226 227 234 304 326 329 333 372 375 378 379

therefore **misses = 16**

false alarms are the Wickelfeatures 176 307 325 328 425

therefore **false alarms = 5**

Rest of the Wickelfeatures are the matching Wickelfeatures.

therefore, **matches = 43**

$$\text{percentage match} = \frac{43}{43 + 16 + 5} * 100 = 67\%$$

Similarly, for camed ,

matches = 38
misses = 21

false alarms = 26
so, percentage match = 44%

and for comed ,

matches = 82
misses = 27
false alarms = 32
so, percentage match = 35%.

In the following paragraphs, the results of simulation for learning the past tense are presented.

1. The model demonstrates similar 3-stage learning as children do. In table 4.1, we present the model's response for the different verbs in the three phases. Alternatives having the maximum percentage match are highlighted. We note that in phase-1 for all the ten verbs (which were used for training), the correct past tense form has 100% match. Hence model always gets the correct past tense for the verbs.

In phase-2, percentage match for irregular past tense form starts decreasing whereas for regularized alternatives percentage match increases. Although for most of the verbs, the match for correct irregular past tense form is still much greater than that of incorrect regularized alternatives so that the model will have no difficulty in responding with correct alternatives. For two verbs viz. come (came 42%, comed 33% and camed 48%) and make (made 55% and maked 50%) correct irregular alternative and incorrect regularize alternatives have comparable matches. Hence the model has tendency to use incorrect alternatives with almost equal likelihood.

Note that for the regular verbs live and look model's response has not changed at all. They are still 100%

This is in accordance with the phenomenon reported by Kuczaj [1977].

In phase-3, when the model has seen enough cycles (in our case 200) of all the regular and irregular verbs, it adjusts the connection strengths so that it can respond with correct alternatives for past tense form.

2. System's response is excellent in stage-3. It generates correct output for almost all the verbs it was taught. In addition, even for new verbs, it is able to generate correct past tense form most of the time. In table 4.2, we compare the percentage of Wickelfeatures matched for the correct response and the best incorrect response (by best incorrect response, we mean the alternative for which the percentage match is maximum among all incorrect alternatives). We find that match for the correct alternative is usually much greater than the incorrect alternatives, so that it will be able to respond with the right alternatives most of the time.

3. It can be said that the model exhibits the learning of the past tense pretty well. However, for some verbs like hide, wear etc, the model generates past tense in regular as well as irregular form with almost equal probability.

Is it that the model is not able to learn tha past tense formation ?

Before making any conclusive remark, we should note that even we do exhibit similar behaviour sometimes. How else can we explain regularized ás wel as irregular allternatives for some of the verbs e.g. spoil -- spoiled, spoilt; speed -- sped, speeded; hang -- hung, hanged etc. Incidentally, the model also has almost

TABLE 4.1

Percentage match in Wickelfeatures for different alternatives in the three phases

verb	past tense form	phonetic spelling of past tense form	phase 1	phase 2	phase 3
come	came	/kAm/	100	42	67
	comed	/kxmd/	17	48	35
	camed	/kAmd/	51	33	44
look	looked	/lukt/	100	100	100
get	got	/got/	100	95	93
	get	/got/	60	59	53
give	gave	/gAv/	100	62	77
	gived	/givd/	23	45	36
	gaved	/gAvd/	48	53	53
take	took	/tuk/	100	64	72
	taked	/tAkt/	17	28	22
go	went	/went/	100	92	96
	wented	/wented/	42	42	46
have	had	/had/	100	59	83
	haved	/havd/	27	44	30
live	lived	/livd/	100	100	100
feel	felt	/felt/	100	78	95
	feeled	/fEld/	39	50	42
make	made	/mAd/	100	55	75
	maked	/makt/	27	50	34

equal percentage match for **sped** and **speeded** as past tense of **speed**.

4. System responds with double past tense for some of the verbs e.g. drive--droved. In general native speakers of English do not make such mistakes but sometimes they too do. For example, lend-- lented.

Looking at the table of percentage match for the past te forms of different verbs, we find that for most of the verbs the system has excellent match for the correct alternatives. Whenever an incorrect alternative has significant match, the match for correct alternative in most cases is much greater than that of incorrect alternative so that the system will have no difficulty in responding with correct alternatives. Hence,in brief, we can claim that the model is able to learn the rules of transformation to form the past tense form from a given verb.

For many verbs, **past + ed** form has significant match. For example, **wear -- wore /wOr/, wored /wOrd/**. This is inevitable because **/wOrd/** already has 3 matching Wickelphones with **/wOr/**. The fact that **/wOr/** still has a better match than **/wOrd/** is sufficient to show that the model is computing the correct response.

TABLE 4.2

System's response for some of the verbs

verb	phonetic input	'phonetic spelling of past tense form	past tense form	percentage match in Wickelfeatures
keep	kEp	kept	kept	96
		kEpt	keeped	46
sleep	slEp	slept	slept	93
		slEpt	sleeped	67
drive	drIv	drOv	drove	81
		drIved	drived	62
		droved	droved	66
send	send	sent	sent	98
build	bild	bilt	built	100
		bilded	builded	48
know	nO	nyu	knew	97
throw	TrO	Tryu	Threw	96
		TrOd	Throwed	25
draw	drq	dryu	drew	92
		drqd	drawed	39
wear	wExr	wOr	wore	74
		wOrd	wored	67
		wExrd	weared	36
hide	hId	hid	hid	56
		hIded	hided	52
		hided	hided	46
burn	bxrn	bxrnt	burnt	93
spin	spin	spxn	spun	87
		spind	spinned	54
		spxnd	spunned	64
read	rED	red	read	97
find	fInd	fWnd	found	78
		fInded	finded	46
laugh	lXf	lXft	laughed	100
kiss	kis	kist	kissed	96
love	lxv	lxvd	loved	98
fire	fIr	fIrd	fired	100
desire	diZIr	diZIrd	desired	100
impress	impres	imprest	impressed	100
subscribe	sxbskrIb	sxbskrIbd	subscribed	100
suck	sxk	sxkt	sucked	100
open	opxn	opxnd	opened	100
provoke	prxvOk	prxvOkt	provoked	100
eat	Et	At	ate	66
		Eted	eated	50
fly	flI	flyu	flew	95
		flId	flied	36
sing	siN	siNd	singed	17
		saN	sang	100
sting	stIN	stxN	stung	100
dig	dig	dxg	dug	97

TABLE 4.3

Model's response for novel verbs

verb	phonetic input	phonetic spelling of past tense form	past tense form	percentage match in Wickelfeatures
swear	swAxr	swOr	swore	43
	swAxr	swArd	sweared	74
think	Tink	Tot	thought	49
		Tank	thank	26
		tinkt	thinked	32
spread	spred	spred	spread	71
		spreded	spread	63
bully	buli	bulEd	bullied	74
ponder	pondxr	pondxrd	pondered	77
appear	apExr	apExrd	appeared	66
board	bOrd	bOrded	boarded	63
wonder	wxndxr	wxndxrd	wondered	85
abhor	abhxr	abhxrd	abhored	62
blast	blXst	blXsted	blasted	83
dart	dXrt	dXrted	darted	64
drain	drAn	drAnd	drained	94
gear	gExr	gExrd	geared	71
	gExr	gOr	gore	26
	gExr	gOrd	gored	53
light	lIt	lIted	lighted	96
tremble	trembl	trembld	trembled	91
inquire	inkwIr	inkwIrd	inquired	95

CHAPTER 5

SIMULATING SANDHI RULES

To simulate Sandhi rules using the connectionist model, we cannot use the Wickelfeature coding scheme due to the following reasons:

1. There are large number of phonemes in Indian languages, since each letter in the alphabet is a phoneme. Hence the number of Wickelfeatures (or whatever we decide to call them) becomes too large to handle.

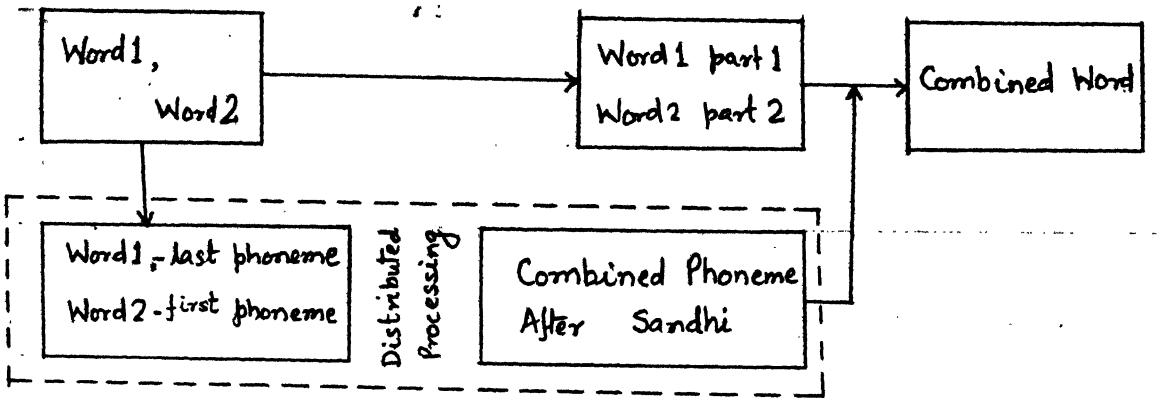
2. Sandhi is a phenomenon limited to word boundaries only. Hence it is extremely wasteful to code the whole word into Wickelfeatures, then perform the sandhi and decode that to get the word.

3. Even if we were able to handle the large number of Wickelfeatures, it is not proper to code the words in Indian languages into Wickelfeatures as quite frequently we find repeating Wickelphones in Indian languages, e.g. /madada/. Anupras Alankar in Indian poetry precisely exploits repeating phonemes to beautify the verses. Hence Wickelfeature coding scheme is inadequate for Indian languages.

We used a highly simplified coding scheme which is described below:

We extract the last phoneme from the preceding word and the first phoneme form the succeding word. Only these two phonemes are processed by the pattern associator network to performk

sandhi. Our model for performing sandhi will look like the following:



Each phoneme is categorized on two dimensions as shown in table 5.1. In addition to the two features associated with a phoneme, the phoneme has a feature which tells whether the phoneme is the first phoneme or the second phoneme. Hence a total of 100 features are used to represent a phoneme.

Learning procedure is the same as mentioned in chapter 3.

TABLE 5.1

CATEGORIZATION OF PHONEMS ON TWO DIMENSIONS

	first	second	third	fourth	nasal
kantha	क	ख	ग	घ	ङ
talavya	च	छ	ज	झ	ञ
moordhanya	ट	ठ	ड	ढ	ण
danta	त	थ	द	ध	न
oshtha	प	फ	ब	भ	म
antah-stha	य	र	ल	व	
ooshma	श	ष	स	ह	

hraswa-swar	अ	इ	उ	ऊ	
dirgha-swar	आ	ई	ऊ		
composite-swar	ए	ऐ	ओ	औं	अं

---

key:

a:अ A:आ i:इ I:ई u:उ U:ऊ R:ऋ e:ए E:ऐ o:ओ O:औं M:अं

k:क K:ख g:ग G:घ c:च C:छ j:ज J:झ t:ट T:ठ

d:ड D:ढ N:ण w:त W:थ q:द Q:ध n:न p:प P:फ b:ब B:भ

m:म y:य r:र l:ल v:व S:श x:ष s:स h:ह

We presented the model with examples of different type of sandhi rules for 25 cycles. Model's response after even 25 cycles was excellent. The model learnt the sandhi rules so fast because there is no exception in the sandhi rules. Hence the model doesn't have to worry about attaining connection strengths that would take care of the exceptions.

A sample run of performing sandhi is given in table 5.2.

rAma	+	ASraya	= rAmASraya
waWa	+	ukwa	= waWokwa
ut	+	Sixta	= ucCixta
maha	+	inqra	= mahenqra
purux	+	uwwama	= puruxowwama
eka	+	eka	= ekEka
maha	+	OxaQi	= mahOxaQi
sUrya	+	aswa	= sUryAswa
parama	+	AwmA	= paramAwmA
viqyA	+	aByAs	= viqyAByAs
aBi	+	Ixta	= aBIxta
rajanI	+	ISa	= rajanISa
nava	+	UDA	= navoDA
sapwa	+	Rxi	= sapwarxi
maha	+	ojas	= mahOjas
naqi	+	ambu	= naqyambu
su	+	Agawa	= svagawa
piwR	+	AjnA	= piwrAjnA
vak	+	ISa	= vagISa
qiw	+	gaja	= qiggaja
uw	+	QaraNa	= uqQaraNa
jagaw	+	nAW	= jagannAW
wejas	+	maya	= wejomaya
wapas	+	carya	= wapaScarya
nis	+	Cala	= biSCala
nis	+	kAma	= nixkAma
nis	+	Pala	= nixPala
nis	+	pakxa	= nixpakxa
ut	+	Svasa	= ucCvasa
sat	+	jana	= sajjana
sat	+	jana	= sajjana

CHAPTER 6

CRITICISMS AND DISCUSSION

Pinker and Prince [1988] have analyzed the linguistic and developmental assumptions of the connectionist model for learning the past tense and criticized the model on the following grounds:

1. It cannot represent certain words.

   As already pointed out in chapter 3, Wickelphones (and hence Wickelfeatures too) are inadequate for coding arbitrary strings of phonemes. In case of repeating Wickelphones in a word, it cannot properly code that word. For example, Pinker and Prince quote two words from an Australian language Oykangand:

   algal (#al alg lga gal al#)
   algalgal (#al alg lga gal alg lga gal al#)

   Set of Wickelfeatures activated by both the words are the same. Hence they are not distinguishable by Wickelfeatures. Similar situation arises in Hindi:

   mada (#ma mad ada da#)
   madada (#ma mad ada dad ada da#)

2. PDP model cannot learn many rules.

3. PDP model can learn rules found in no human language.

   A quintessential unlinguistic map is relating a string to its mirror image reversal. Although neither physiology nor physics forbids it, no language uses such a pattern. Relating mirror image reversal to a word is as easy to learn in the RM model as the identity map.

4. It cannot explain morphological and phonological

regularities.

As already pointed out in chapter 3, Wickelphone representation does not provide a basis for generalizations to emerge about what aspects of the present tense correspond to what aspect of the past tense.

Also, in Wickelphone representation, /#lit/ and /#ilt/ do not have any common Wickelphones. The implicit claim is that such pairs have no phonological properties in common. However, we know that in all the natural languages changes of the type /#ilt/ --> /#lit/ or /#lit/ -- /#ilt/ , based on phonetic similarity are quite common. For example, in the history of English, there are **hross** --> **horse**, **brid** --> **bird** [ref 8]. In Hindi there are whole lot of words (tatsam to tadbhava) where we find similar changes. For example **soorya** --> **sooraj**, **dharitri** --> **dharati** etc. In Wickelphone-Wickelfeature representation, changing **dharitri** to **dharati** is no more likely and no easier than any other complete replacement like **dhanvan**. This is very unsatisfactory.

5. It cannot explain the difference between regular past tense form and irregular past tense form.
6. It cannot handle the elementary problem of homophony.
7. It makes errors in computing the past tense forms of a large percentage of the words it is tested on.

However, we should note that most of the problems mentioned above are due to the Wickelphone-Wickelfeature coding rather than

the connectionist architecture. In particular objections 1,4 and 6 only point out the inadequacy of the Wickelfeature coding scheme. Probably, a better coding scheme would be devised which will be in close resemblance with the way words are coded in the human mind, and which will take care of the above objections.

Second and third objections also are not very serious. Although large number of arbitrary associations (hence large number of rules) cannot be learnt by a pattern associator network, we never find such a case in any realistic situation.

Also, it is true that PDP model can easily learn to associate mirror image reversal of a word to itself. But the model is never going to learn this rule unless it is made to do so.

It is not true that the model makes errors in computing past tense forms of a large percentage of the words it is tested on, as claimed by Pinker and Prince [1988]. For most of the regular verbs, the model generates the correct past tense form. For the irregular verbs used in the learning, the model computes the correct past tense form. Only for the novel irregular verbs, the model commits errors.

The greatest advantage of the PDP model seems to be its ability to exhibit rule based behaviour without actually storing the rules in explicit form. As human beings too learn rule based behaviour without consciously being aware of the rules, it seems that the PDP model might offer a good model of learning. Let us consider the success and failures of PDP model in the context of language acquisition.

In the studies conducted so far, the PDP model has not been able to show any edge over the traditional explicit inaccessible rule view of language acquisition. We will not discuss PDP model vis-a-vis explicit inaccessible rule view, but only the successes and failures of the PDP model.

In the illustrative examples of learning the past tense formation rules of English verbs and sandhi rules in Hindi, the model exhibits considerable success. Hence we might be optimistic about the model's success in other cases also.

A possible objection against the model may be about the size of the matrix of pattern associator (henceforth, we shall call PA-matrix). Complexity of past tense formation rules is almost insignificant compared to the complexity of language acquisition as a whole. But we find that even to perform such a small task, and that too without any finesse, size of the matrix is enormous - a matrix of size 460x460, having more than 200,000 entries. It seems highly unlikely that such an inefficient process is being adopted by human mind.

We know that human mind has excellent pattern recognition capabilities. Behind the language acquisition process also, there lies this ability of pattern recognition. It is highly likely that we have this uncanny pattern recognition capabilities only because information is very efficiently coded inside our brain. Therefore, the size of the PA-matrix will be greatly reduced if inputs and outputs were coded as efficiently as is done inside our brain.

The PDP model ,as it stands now, has other serious limitations:

1. The model treats occurrences of events which conform to the rules as well as those which are exceptions to the rules, uniformly. This poses serious limitations to the model's ability to accommodate exceptions. To accommodate a few exceptions, large number of changes need to be made in the PA-matrix.

If we look at the weights adjustment process in our model, we find that each event locally modifies the weights. Hence, to achieve the final set of weights to accommodate any exception, the model needs to scan many times, the whole set of events it has seen so far. Whereas human beings smoothly learn a few exceptions.

2. Rules are almost same for making past tense and participle forms from present tense forms. Outside the verbal system, there is yet another phenomenon that uses similar morphisms (t-d-ed suffix), viz. making adjectives from nouns. For example

t	d	ed
hooked	long-nosed	one-handed
pimple-faced	horned	talented
thick-necked	winged	strong-headed etc.

Hence, inside our mind, it is likely that these phenomena would be treated alike. But the PDP model would have to maintain different PA-matrices for different phenomena.

3. The learning procedure for the PDP model is not satisfactory. A constant modification in the weights for all the event is contrary to reality.

4. The PDP model is more like a static model as the number of

input units and output units are not changed in the learning process. Even if something like the PDP model were in close resemblance with our learning, the model must be able to update the number of input and output units during the learning process.

5. For a complex phenomenon like language acquisition, there are lot of sub-phenomena viz. covering the rules of orthography, etymology, syntax, punctuation and semantics. For each of these sub phenomenon, there will be PA-matrices. A complex combination of all these sub-models will be needed to properly model the language acquisition. The complexity of this task right now makes it a practical impossibility.

Hence we can conclude that the form of the PDP model,as it stands now is highly inadequate for simulating the language acquisition process. Nevertheless, the PDP model is useful as it gives a new direction for the modeling of learning.

REFERENCES

1. Rumelhart D.E. & McClelland J.L. (1986). On learning the past tenses of English verbs.In **Parallel Distributed Processing. Volume 2:** Psychological and Biological models. Cambridge MA: Bradford books/MIT Press.

2. Pinker S. & Prince A. (1988). On language and connectionism: Analysis of a Parallel distributed processing model of language acquisition. In **Cognition, 28(1988) 73-193**

3. Erwin S. (1964). Imitation and structural change in children's language. In **E. Lenneberg (Ed)** :New directions in the study of language. Cambridge, MA: MIT Press.

4. Brown R. (1973). **A first language**. Cambridge, MA: Harvard University Press.

5. Kuczaj S.A. (1977). The acquisition of regular and irregular past tense forms. **Journal of Verbal Learning and Verbal Behavior, 16, 589-600.**

6. Wickelgren W.A. (1969). Context-sensitive coding, associative memory, and serial order in (speech) behavior. **Psychological Review, 76, 1-15.**